Yee & Associates, P.C.

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

Facsimile Cover Sheet

To: Commissioner for Patents for **Examiner Emmanuel Coffy** **Group Art Unit 2157**	Facsimile No.: **571/273-8300**
From: Jennifer Pilcher Legal Assistant to Wayne Bailey	No. of Pages Including Cover Sheet: **29**

Message:

Enclosed herewith:

- Transmittal Document; and
- Appeal Brief.

Re: Application No. **09/864,117**
Attorney Docket No: **AUS920010285US1**

Date: Tuesday, September 06, 2005

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Sedlack**

Serial No.: **09/864,117**

Filed: **May 24, 2001**

For: **Method and Apparatus to Solve Compatibility Between Heterogeneous Web Server Access Logs Formats**

Group Art Unit: **2157**

Examiner: **Coffy, Emmanuel**

Attorney Docket No.: **AUS920010285US1**

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on September 6, 2005.

By: Jennifer Pilcher
Jennifer Pilcher

TRANSMITTAL DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Appeal Brief (37 C.F.R. 41.37).

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

Respectfully submitted,

Duke W. Yee
Registration No. 34,285
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777
ATTORNEY FOR APPLICANT

Docket No. AUS920010285US1

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Sedlack**

Serial No. **09/864,117**

Filed: **May 24, 2001**

For: **Method and Apparatus to Solve Compatibility Between Heterogeneous Web Server Access Logs Formats**

Group Art Unit: **2157**

Examiner: **Coffy, Emmanuel**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on September 6, 2005.

By: Jennifer Pilcher
Jennifer Pilcher

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on July 5, 2005.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

09/07/2005 CCHAU1 00000063 090447 09864117
01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-18

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: none
2. Claims withdrawn from consideration but not canceled: none
3. Claims pending: 1-18
4. Claims allowed: none
5. Claims rejected: 1-18
6. Claims objected to: none

C. CLAIMS ON APPEAL

The claims on appeal are: 1-18

STATUS OF AMENDMENTS

No amendment after final was filed for this case.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

Server computer systems such as web servers maintain log files or logs which record (i) incoming requests and (ii) attempts to gain access to the servers. These access logs can be saved in various dissimilar/incompatible formats. The present invention is directed to a technique for converting dissimilar log file formats into a common log file format without requiring specific or special code for each type of log file format. Specifically, Claim 1 is directed to a method for establishing compatibility between heterogeneous web server access log formats. A description of an access log file of a web server is supplied by opening a customizable configuration file and, if the access log is static, a log pattern definition which describes data elements, order, and syntax of log entries is set. If the access log is dynamic, a dictionary feature for a log pattern definition is set. A computer process is invoked, where the computer process in turn invokes a translation engine which translates the described web server access log file to a desired log format and returns the translated file back to the computer process (Specification page 14, last two paragraphs and page 15, first paragraph (describing Figure 6); Figure 6, all elements).

B. CLAIM 6 - INDEPENDENT

Claim 6 is directed to a method for translating heterogeneous web server access log formats, and is specifically directed to a process for converting data in a log file to a common log format. A customizable configuration file is read and configuration objects are created. Data in a server access log file is translated line-by-line. For a given line, it is parsed and if there is no error in retrieving required data, both the data elements and syntax are reordered to mimic the desired log format. If there is an error in retrieving required data, a check is made to determine if there is an entry in the web server comment field. If there is no entry in the comment field, the file line is reread. If there is an entry in the comment field, a check is made to ensure that a dictionary feature is enabled. The translated data is sent to a computer process for subsequent processing and the process ends when all lines in the access log file have been read (Specification page 19, last paragraph – page 20, first paragraph (describing Figure 9); Figure 9, all elements; page 22, top of page – page 23, middle of page (describing Figure 12); Figure 12, all elements).

C. CLAIM 11 - INDEPENDENT

Claim 11 is a computer program product claim of similar scope to Claim 1, and the summary of Claim 1 given above is equally applicable to Claim 11, and is thus hereby incorporated by reference in order to provide the summary of Claim 11.

D. CLAIM 12 - INDEPENDENT

Claim 12 is a computer program product claim of similar scope to Claim 6, and the summary of Claim 6 given above is equally applicable to Claim 12, and is thus hereby incorporated by reference in order to provide the summary of Claim 12.

E. CLAIM 17 - INDEPENDENT

Claim 17 is a system claim of similar scope to Claim 1, and the summary of Claim 1 given above is equally applicable to Claim 17, and is thus hereby incorporated by reference in order to provide the summary of Claim 17.

F. CLAIM 18 - INDEPENDENT

Claim 18 is a system claim of similar scope to Claim 6, and the summary of Claim 6 given above is equally applicable to Claim 18, and is thus hereby incorporated by reference in order to provide the summary of Claim 18.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 1-18)

Claims 1-18 stand rejected under 35 U.S.C. § 103 as being unpatentable over Nareddy et al. (US 6,785,666) in view of Nair et al. (US 6,741,990).

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-18)

A.1. Claims 1, 2, 5, 11, 17

Claim 1 provides as follows:

> A method for establishing compatibility between heterogeneous web server access log formats, comprising:
>
> (1) supplying a description of an access log file of a web server, by: (i) opening a customizable configuration file; (ii) *if the access log is static*, setting a log pattern definition to describe data elements, order, and syntax of log entries; (iii) *if the access log is dynamic*, setting a dictionary feature for a log pattern definition; (iv) saving and exiting the configuration file; and
>
> (2) invoking a computer process, wherein the process in turn invokes a web server access log translation engine (WSALTE) which translates the described web server access log file to a desired log format and returns the translated file back to the computer process. (emphasis and numerics added for reading clarity)

As can be seen, Claim 1 is directed to a method for establishing compatibility between heterogeneous web server access log formats. Appellants urge that Nareddy shows a method of providing customers with access to data from web logs. Nareddy also shows a method of reformatting the data in order allow a support system application to more easily use the data. Importantly, Nareddy assumes use of a *single, common format* for the log files that are processed (col. 7, lines 56-61), and if a log file does not conform to the expected format, it is rejected. As described by Nareddy at col. 7, lines 19-22:

> The parser includes a filter log entry component 311, a normalize log entry component 312, a generate dimensions component 313, an identify sessions component 314, and a generate aggregate statistics component 315. The filter log entry component identifies which log entries should not be included in the main data warehouse. *For example, a log entry that has*

> *an invalid format should not be included.* The normalize log entry component normalizes the data in a log entry

Nareddy explicitly provides that a log entry having an invalid format should *not* be included in the filtered log. Thus, all references that Nareddy makes regarding reformatting information in the log has nothing to do with *establishing compatibility between heterogeneous web server access log formats*, as expressly recited in Claim 1. As explained above, Nareddy is incapable of doing so, as it relies upon a single, predetermined format. Nair, which shows a method of excluding IP addresses from a web log that can already be read, fails to cure the lack of disclosure in Nareddy. Thus, the proposed combination does not teach or otherwise suggest the invention recited in Claim 1 of a method for establishing compatibility between heterogeneous web server access log formats.

Still further with respect to Claim 1, such claim recites conditional operations that are performed *based upon whether the access log is static or dymamic.* In particular, Claim 1 recites: *if the access log is static*, setting a log pattern definition to describe data elements, order, and syntax of log entries; and *if the access log is dynamic*, setting a dictionary feature for a log pattern definition. In rejecting the static access log aspect of this claim, and its associated processing, the Examiner cites Nareddy's teachings at col. 5, lines 5-8, and in rejecting the dynamic access log aspect of this claims, and its associated processing, the Examiner cites Nareddy's teachings at col. 6, lines 42-56. Appellants urge multiple errors in such assertion, as now described in detail.

The cited passage at Nareddy col. 5 is with respect to fact tables and dimension tables that represent high-level facts and attributes derived from the low-level facts and attributes of log files (col. 4, lines 64-67). Since high-level facts and attributes may not be derivable from the data in a single log entry, an example is given of where a higher level category of a web page may be identified using a mapping of web page URIs to categories, and that these categories may be stored in a category dimension table. It is also stated that certain facts may only be derivable by analyzing a series of log entries. This passage does not describe any conditional operation being performed based upon whether an access log is static, and it therefore follows that this cited passage does not teach or otherwise suggest the specific claimed feature of setting a log pattern definition to describe (i) data elements, (ii) order, and (iii) syntax of log entries *if the access log is static.*

The cited passage at Nareddy col. 6 makes a generalized statement regarding a parser that analyzes low-level events of customer data and identifies high level events, and converts the customer data into a format that facilitates processing by the decision support system application. It is initially noted that this description of converting low-level events to high level events is the same conversion that is described in the cited passage at col. 5 that was used in rejecting the static access log aspect of Claim 1. Thus, the combination of the passage at col. 5 and the passage at col. 6 does not teach two different operations being performed based upon whether the access log is static or dynamic, as the two passages describe that same thing – albeit one description being more embellished than the other. Importantly, this passage cited at col. 6 makes no mention of any type of determination of whether the access log is dynamic, and thus it necessarily follows that this cited passage does not teach or otherwise suggest the specific claimed step of "if the access log is dynamic, setting a dictionary feature for a log pattern definition", as expressly recited in Claim 1. The examiner cites Nair for the proposition that Nair teaches a dynamic log file where blocks of IP addresses are to be filtered, citing Nair as follows:

> Given the list of exclusion IP addresses to filter, an optimum algorithm needs to be selected for searching in the list of exclusion IP addresses for matches. As mentioned previously, the list of accesses to be filtered, or the list of exclusion IP address in this case, often varies from one client web server to another because of different performance analysis or different filtering requirements exist at different client web servers. For example, each client may decide to filter activities coming from its own company, so the list of exclusion IP address in each case is different. To reduce the cost of identifying whether a particular IP address in the web log file is to be ignored, the adaptive process explores the performance of several different algorithms and data structures. In block P310, metrics are generated for each algorithm to calculate the effectiveness of the algorithm for the list of exclusion IP address. The algorithms is chosen from any number of predefined methods, including, but not limited to, binary search, multi-level dynamic table indexing, adaptive hashing, and bit pattern based exclusion.
>
> The interaction of these algorithms with the IP addresses from the web log file (block P200) will be explained in more detail later, as the discussion now centers on computing the metrics for each algorithm. In one implementation, the metrics to be used to identify the optimum algorithm for a list of exclusion IP addresses are the number of exclusion IP addresses to be filtered and the number of unique combinations to be

> filtered. For example, binary search algorithm is often employed if there is a relative small number of individual exclusion IP addresses to filter. However, binary search algorithm is probably not the best algorithm if you have blocks of IP addresses to filter. In case of binary search, the number of exclusion IP addresses to filter may be one of the determinative metrics, and as the number grows the performance decreases.
>
> Multi-level dynamic table indexing algorithm is suitable when there are blocks of IP addresses to filter. The metric in this case would be the number of unique combinations of octets in the list of exclusion IP addresses, assuming that eight bits are used for indexing.

The cited text does not teach or suggest using a dictionary definition *if the access log is dynamic.* Claim 1 expressly recites such conditional operation *based upon whether the access log is dynamic.*

Importantly, because the cited Narrady reference only contemplates use of a single, predetermined format for data collected from customer servers, as described above, there would have been no reason or motivation to modify the teachings of such reference to include the static/dynamic access log conditional processing, as per the present invention, as the log file format is predetermined (see, e.g., Nareddy col. 7, lines 56-61). The fact that a prior art device could be modified so as to produce the claimed device is not a basis for an obviousness rejection unless the prior art suggested the desirability of such a modification. *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984). Although a device may be capable of being modified to run the way [the patent applicant's] apparatus is claimed, there must be a suggestion or motivation *in the reference* to do so. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). There is simply no desire, suggestion, or other motivation to modify the teachings of the cited references in accordance with the claimed invention, further evidencing non-obviousness of the present invention.

It is therefore further shown that the Examiner has failed to properly establish a prima facie showing of obviousness with respect to Claim 1[1], and thus Claim 1 has been erroneously

[1] To establish prima facie obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. MPEP 2143.03. *See also, In re Royka*, 490 F.2d 580 (C.C.P.A. 1974).

rejected under 35 USC 103[2]. In addition, since a prima facie case of obviousness has not been properly established, the burden has not shifted to Appellants to rebut the obviousness assertion[3].

A.2. Claim 3

Regarding Claim 3, this dependent claim is non-obvious in view of the cited references for the same reasons given above with respect to Claim 1. In addition, Claim 3 contains additional patentable features not taught or suggested by the combination of Nareddy and Nair. For example, neither reference teaches or suggests the claimed log pattern definition feature of "if a data element contains a delimiter that may exist in another data element, isolating the data by replacing the WSALTE terminology with a user substitute definition". In rejecting this aspect of Claim 3, the Examiner states:

> "Nareddy fails to specifically teach equating user substitution definition. However, Nair teaches algorithm for searching and replacing user definition".

Appellants urge that even if such assertion were true, such general assertion does not establish a teaching or suggestion of the specific features recited in Claim 3. In particular, a teaching of an algorithm for searching and replacing user definition does not establish a teaching or suggest of "*if a data element contains a delimiter that may exist in another data element*, isolating the data by replacing the WSALTE terminology with a user substitute definition". This conditional replacement step is not taught or suggested by any of the cited references, nor has the Examiner alleged any such teaching or suggestion, and therefore the Examiner has failed to properly establish a prima facie showing of obviousness, as <u>all</u> claimed features are not taught or suggested by the cited references, MPEP 2143.03. *In re Royka*, supra. The rejection of Claim 3 is thus shown to be in error.

---

[2] If the examiner fails to establish a prima facie case, the rejection is improper and will be overturned. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988).

[3] In rejecting claims under 35 U.S.C. Section 103, the examiner bears the initial burden of presenting a prima facie case of obviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if that burden is met, does the burden of coming forward with evidence or argument shift to the applicant. *Id.*

A.3. Claim 4

Regarding Claim 4, this dependent claim is non-obvious in view of the cited references for the same reasons given above with respect to Claim 1. In addition, Claim 4 contains additional patentable features not taught or suggested by the combination of Nareddy and Nair. For example, neither reference teaches or suggests the claimed dictionary setting feature of "if the name of the data element equates with a WSALTE name in the dictionary feature, equating the server name with a WSALTE name; if the name of the data element does not equate with a WSALTE name, determining if the data element name contains multiple WSALTE names; if the data element name contains multiple WSALTE names, providing substitute definitions, using WSALTE names, which describe discreet data elements; and if the data element name does not contain multiple WSALTE names, equating the server name with an ignore label". In rejecting Claim 4, the Examiner states:

> "Nareddy teaches examining each of a plurality of data elements in an entry of the access log file and removing non-unique identifiers. Nareddy fails to specifically teach equating user substitution definition. However, Nair teaches algorithm for searching and replacing user definition."

Appellants urge that even if such assertion were true, such general assertion does not establish a teaching or suggestion of the specific features recited in Claim 4. In particular, a teaching of an algorithm for searching and replacing user definition does not establish a teaching or suggest of "if the name of the data element equates with a WSALTE name in the dictionary feature, equating the server name with a WSALTE name; if the name of the data element does not equate with a WSALTE name, determining if the data element name contains multiple WSALTE names; if the data element name contains multiple WSALTE names, providing substitute definitions, using WSALTE names, which describe discreet data elements; and if the data element name does not contain multiple WSALTE names, equating the server name with an ignore label". These conditional steps are not taught or suggested by any of the cited references, nor has the Examiner alleged any such teaching or suggestion, and therefore the Examiner has failed to properly establish a prima facie showing of obviousness, as all claimed features are not

taught or suggested by the cited references, MPEP 2143.03. *In re Royka*, supra. The rejection of Claim 4 is thus shown to be in error.

A.4. Claims 6, 12 and 18

With respect to Claim 6, none of the cited references teach or suggest the claimed data translation feature of "parsing a server access log file line; if there is no error in retrieving required data, reordering data elements and syntax to mimic the desired log format; if there is an error in retrieving required data, checking if there in an entry in the web server comment field; if there is no entry in the comment field, rereading the file line; if there is an entry in the comment field, checking that a dictionary feature is enabled". In rejecting Claim 6, the Examiner solely relies upon the same reasoning given in rejecting Claims 1-5. Appellants urge that as none of Claims 1-5 recite the missing claimed feature identified above, the Examiner has failed to establish, or even allege, a prima facie showing of obviousness. Therefore, Claim 6 has been erroneously rejected and the burden has not shifted to Applicants to rebut such obviousness assertion.

A.5. Claims 7 and 13

With respect to Claim 7, none of the cited references teach or suggest the claimed log pattern variable setting feature of "if the definition of the log pattern is static, parsing the log pattern definition and reading a data element; if the data element does not correspond to a web server log translation engine (WSALTE) name, calling a protocol which resolves naming convention; if the data element does correspond to a WSALTE name, setting a WSALTE variable to the name of the data element position". In rejecting Claim 7, the Examiner solely relies upon the same reasoning given in rejecting Claims 1-5. Appellants urge that as none of Claims 1-5 recite the missing claimed log pattern setting feature identified above, the Examiner has failed to establish, or even allege, a prima facie showing of obviousness. Therefore, Claim 7 has been erroneously rejected and the burden has not shifted to Applicants to rebut such obviousness assertion.

A.6. Claims 8 and 14

With respect to Claim 8, none of the cited references teach or suggest the claimed dictionary checking feature of "if the dictionary feature is not enabled, reading the next line in the access log

file; if the dictionary feature is enabled, determining if the comment field entry describes a new log pattern; if the comment field entry does not describe a new log pattern, reading the next line in the access log file; and if the comment field entry does describe a new log pattern, translating the entry in the comment field to WSALTE terminology". In rejecting Claim 8, the Examiner solely relies upon the same reasoning given in rejecting Claims 1-5. Appellants urge that as none of Claims 1-5 recite the missing claimed dictionary checking feature identified above, the Examiner has failed to establish, or even allege, a prima facie showing of obviousness. Therefore, Claim 8 has been erroneously rejected and the burden has not shifted to Applicants to rebut such obviousness assertion.

In conclusion, Appellants have shown numerous errors in the Examiner's final rejection of Claims 1-18, and accordingly requests that this Board reverse such rejections.

Duke W. Yee
Reg. No. 34,285
Wayne P. Bailey
Reg. No. 34,289
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method for establishing compatibility between heterogeneous web server access log formats, comprising:

supplying a description of an access log file of a web server, by:

opening a customizable configuration file;

if the access log is static, setting a log pattern definition to describe data elements, order, and syntax of log entries;

if the access log is dynamic, setting a dictionary feature for a log pattern definition;

saving and exiting the configuration file; and

invoking a computer process, wherein the process in turn invokes a web server access log translation engine (WSALTE) which translates the described web server access log file to a desired log format and returns the translated file back to the computer process.

2. The method according to claim 1, wherein the computer process comprises at least one of a tool, application, and adapter.

3. The method according to claim 1, wherein the step of setting the log pattern definition for a static access log further comprises:

examining each of a plurality of data elements in an entry of the access log file;

if a data element contains a delimiter that may exist in another data element, isolating the data by replacing the WSALTE terminology with a user substitute definition;

equating the user substitute definition with WSALTE terminology;

equating the user substitute definition delimiters with non-unique delimiters; and

removing non-unique delimiters from a parent delimiter list.

4. The method according to claim 1, wherein the step of setting the dictionary feature for a dynamic access log further comprises:

examining each of a plurality of data elements in an entry of the access log file;

if the name of the data element equates with a WSALTE name in the dictionary feature, equating the server name with a WSALTE name;

if the name of the data element does not equate with a WSALTE name, determining if the data element name contains multiple WSALTE names;

if the data element name contains multiple WSALTE names, providing substitute definitions, using WSALTE names, which describe discreet data elements; and

if the data element name does not contain multiple WSALTE names, equating the server name with an ignore label.

5. The method according to claim 1, wherein the customizable configuration file is an ASCII file.

6. A method for translating heterogeneous web server access log formats, comprising:

reading a customizable configuration file and creating configuration objects;

setting a log pattern variable;

opening a server access log file and reading a line within the file;

translating data in the file line into a desired log format, by:

parsing a server access log file line;

if there is no error in retrieving required data, reordering data elements and syntax to mimic the desired log format;

if there is an error in retrieving required data, checking if there in an entry in the web server comment field;

if there is no entry in the comment field, rereading the file line;

if there is an entry in the comment field, checking that a dictionary feature is enabled;

sending the translated data to a computer process; and

exiting when all lines in the access log file have been read.

7. The method according to claim 6, wherein the step of setting the log pattern variable further comprises:

retrieving configuration objects for data elements, order, and syntax of the log pattern;

retrieving configuration objects for a log pattern delimiter list;

if the definition of the log pattern is static, parsing the log pattern definition and reading a data element;

if the data element does not correspond to a web server log translation engine (WSALTE) name, calling a protocol which resolves naming convention;

if the data element does correspond to a WSALTE name, setting a WSALTE variable to the name of the data element position; and

closing the configuration file when all data elements in the log pattern definition have been read.

8. The method according to claim 6, wherein the step of checking that the dictionary feature is enabled further comprises:

if the dictionary feature is not enabled, reading the next line in the access log file;

if the dictionary feature is enabled, determining if the comment field entry describes a new log pattern;

if the comment field entry does not describe a new log pattern, reading the next line in the access log file; and

if the comment field entry does describe a new log pattern, translating the entry in the comment field to WSALTE terminology.

9. The method according to claim 8, wherein the step of translating the entry in the comment field to WSALTE terminology further comprises:

reading a data element in the comment field;

if a WSALTE name equates with the name of the data element, replacing the data element name with the WSALTE name;

if a WSALTE name does not equate with the name of the data element, replacing the data element name with an ignore label; and

exiting after all data elements in the comment field have been read.

10. The method according to claim 6, wherein the customizable configuration file is an ASCII file.

11. A computer program product in a computer readable medium for use in a data processing system, for establishing compatibility between heterogeneous web server access log formats, comprising:

instructions for receiving a description of an access log file of a web server, by:

opening a customizable configuration file;

if the access log is static, setting a log pattern definition to describe data elements, order, and syntax of log entries;

if the access log is dynamic, setting a dictionary feature for a log pattern definition;

saving and exiting the configuration file; and

instructions for invoking a computer process, wherein the process in turn invokes a web server access log translation engine (WSALTE) which translates the described web server access log file to a desired log format and returns the translated file back to the computer process.

12. A computer program product in a computer readable medium for use in a data processing system, for translating heterogeneous web server access log formats, the computer program product comprising:

instructions for reading a customizable configuration file and creating configuration objects;

instructions for setting a log pattern variable;

instructions for opening a server access log file and reading a line within the file;

instructions for translating data in the file line into a desired log format, by:

parsing a server access log file line;

if there is no error in retrieving required data, reordering data elements and syntax to mimic the desired log format;

if there is an error in retrieving required data, checking if there in an entry in the web server comment field;

if there is no entry in the comment field, rereading the file line;

if there is an entry in the comment field, checking that a dictionary feature is enabled;

instructions for sending the translated data to a computer process; and

instructions for exiting when all lines in the access log file have been read.

13. The computer program product according to claim 12, wherein the instructions for setting the log pattern variable further comprises:

instructions for retrieving configuration objects for data elements, order, and syntax of the log pattern;

instructions for retrieving configuration objects for a log pattern delimiter list;

if the definition of the log pattern is static, instructions for parsing the log pattern definition and reading a data element;

if the data element does not correspond to a web server log translation engine (WSALTE) name, instructions for calling a protocol which resolves naming convention;

if the data element does correspond to a WSALTE name, instructions for setting a WSALTE variable to the name of the data element position; and

instructions for closing the configuration file when all data elements in the log pattern definition have been read.

14. The method according to claim 12, wherein the instructions for checking that the dictionary feature is enabled further comprises:

if the dictionary feature is not enabled, instructions for reading the next line in the access log file;

if the dictionary feature is enabled, instructions for determining if the comment field entry describes a new log pattern;

if the comment field entry does not describe a new log pattern, instructions for reading the next line in the access log file; and

if the comment field entry does describe a new log pattern, instructions for translating the entry in the comment field to WSALTE terminology.

15. The computer program product according to claim 14, wherein the instructions for translating the entry in the comment field to WSALTE terminology further comprises:

instructions for reading a data element in the comment field;

if a WSALTE name equates with the name of the data element, instructions for replacing the data element name with the WSALTE name;

if a WSALTE name does not equate with the name of the data element, instructions for replacing the data element name with an ignore label; and

instructions for exiting after all data elements in the comment field have been read.

16. The computer program product according to claim 12, wherein the customizable configuration file is an ASCII file.

17. A system for establishing compatibility between heterogeneous web server access log formats, comprising:

a receiving component which receives a description of an access log file of a web server, by:

opening a customizable configuration file;

if the access log is static, setting a log pattern definition to describe data elements, order, and syntax of log entries;

if the access log is dynamic, setting a dictionary feature for a log pattern definition;

saving and exiting the configuration file; and

an invoking component which invokes a computer process, wherein the process in turn invokes a web server access log translation engine (WSALTE) which translates

the described web server access log file to a desired log format and returns the translated file back to the computer process.

18. A system for translating heterogeneous web server access log formats, comprising:

a reading component which reads a customizable configuration file and creates configuration objects;

a selection component which sets a log pattern variable;

an opening component which opens a server access log file and reads a line within the file;

a translating component which translates data in the file line into a desired log format, by:

parsing a server access log file line;

if there is no error in retrieving required data, reordering data elements and syntax to mimic the desired log format;

if there is an error in retrieving required data, checking if there in an entry in the web server comment field;

if there is no entry in the comment field, rereading the file line;

if there is an entry in the comment field, checking that a dictionary feature is enabled;

a communication component which sends the translated data to a computer process; and

an exiting mechanism which exits when all lines in the access log file have been read.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.